Crowdsourced Disaster Relief Platform Technical Documentation

Course: CS 3354 Spring 2025 Group Number: 2 Group Members: Casey Nguyen, Kevin Pulikkottil, Andy Jih, Sawyer

**Table of Contents:**

1. System Overview
2. Technology Stack
3. System Architecture
4. API Endpoint Details
5. AI Matching Algorithm
6. Database Schema
7. Security and Reliability
8. System Overview

The Crowdsourced Disaster Relief Platform connects victims, volunteers, and NGOs in real-time to streamline relief efforts during disasters. The backend system provides essential functionality through AI-driven volunteer matching, enabling quick and efficient distribution of aid.

1. Technology Stack

Backend:

* FastAPI (Python)
* PostgreSQL
* SQLAlchemy
* scikit-learn (KNN Algorithm)

Frontend:

* Flutter (mobile/web)

Deployment:

* Docker & Docker Compose

1. System Architecture

* Frontend Subsystem (Flutter)
* Backend API Subsystem (FastAPI)
* Database Subsystem (PostgreSQL)
* AI Matching Subsystem (KNN/scikit-learn)

The frontend communicates with the backend via RESTful APIs. The backend interacts with the PostgreSQL database using SQLAlchemy ORM, processes data with the KNN algorithm, and returns JSON responses to the frontend.

[Include architecture diagram here]

1. API Endpoint Details

* GET /match/{request\_id}
  + Description: Retrieves the best volunteer matches based on request type and location.
  + Response Example:

{

"matched\_volunteers": [

{"id": 1, "name": "Alice", "skills": "Medical", "location": "Houston"},

{"id": 2, "name": "Bob", "skills": "Food Logistics", "location": "Austin"},

{"id": 3, "name": "Charlie", "skills": "Rescue", "location": "Dallas"}

]

}

1. AI Matching Algorithm

The matching uses the K-Nearest Neighbors (KNN) algorithm:

* Volunteers and requests are numerically encoded using hash functions on their skills and location attributes.
* KNN finds the nearest volunteers based on Euclidean distance.
* Ensures quick retrieval and accuracy for real-time matching.

1. Database Schema

**Volunteer Table:**

* id (Integer, Primary Key)
* name (String)
* skills (String)
* location (String)

**Request Table:**

* id (Integer, Primary Key)
* type (String)
* location (String)

1. Security and Reliability

* **CORS Enabled:** Ensures safe integration with Flutter frontend.
* **Database Security:** Credentials managed securely through environment variables.
* **Error Handling:** Clear JSON error responses to aid frontend debugging.

Future recommendations include Role-Based Access Control (RBAC) and Two-factor Authentication for improved security.